Probabilistic Model Checking for Systems Biology

Marta Kwiatkowska[1], Gethin Norman[2], and David Parker[1]

[1] Oxford University Computing Laboratory, Parks Road, Oxford, OX1 3QD, UK
[2] Department of Computing Science, University of Glasgow, Glasgow, G12 8RZ, UK

Abstract. Probabilistic model checking is a technique for formally verifying quantitative properties of systems that exhibit stochastic behaviour. In this chapter, we show how this approach can be applied to the study of biological systems such as biochemical reaction networks and signalling pathways. We present an introduction to the state-of-the-art probabilistic model checking tool PRISM using a case study based on the Fibroblast Growth Factor (FGF) signalling pathway.

1 Introduction

Probabilistic model checking is a formal verification technique for the analysis of systems with stochastic characteristics. It is based on the exhaustive construction and analysis of a probabilistic model, typically a Markov chain or Markov process, of some real-life system. This allows various quantitative properties of the system's behaviour, such as performance or reliability, to be analysed automatically under a range of different scenarios or parameters. The results can then be used to identify anomalies, faults or other points of interest in the systems being modelled. Probabilistic model checking has been used to study a wide array of systems including randomised communication protocols such as Bluetooth and Firewire, security protocols for anonymity and contract signing, dynamic power management schemes and NAND multiplexing for nanotechnology.

In this chapter we focus on how probabilistic model checking can be used to study the behaviour of biological systems such as biochemical reaction networks and signalling pathways which, under certain assumptions, are well suited to being modelled as discrete stochastic systems. We give an introduction to probabilistic model checking and to the software tool PRISM [10, 17] which implements these techniques and, using a selection of examples of increasing complexity, demonstrate how PRISM can used as a framework for the modelling and analysis of biological pathways. The approaches introduced in this chapter should not be viewed as a substitute for the classical approaches based on simulation and differential equations. Rather, the techniques should be used in conjunction with these classical approaches to obtain a more detailed understanding of the complex interactions and behaviour of biological pathways.

The chapter is organised as follows. Section 2 presents an overview of probabilistic model checking, including descriptions of continuous-time Markov chains, the temporal logic CSL and the probabilistic model checking tool PRISM. In Sections 3, 4 and 5, we consider three examples to demonstrate both how to model biological pathways in the PRISM language and how probabilistic model checking can be used to analyse such pathways. In Section 3 we start with a simple set of reactions. Then, in Sections 4 and 5, we extend this simple example, first by increasing the number of components of each molecular species in the system and then by extending the possible reactions of the system. Finally, Section 6 outlines related topics of interest.

2 Probabilistic Model Checking and PRISM

Probabilistic model checking is a technique for the modelling and analysis of systems which exhibit stochastic behaviour. This technique is a variant of *model checking*, a well-established and widely used formal method for ascertaining the correctness of real-life systems. Model checking requires two inputs: a description of the system in some high-level modelling formalism (such as a Petri net or process algebra), and specification of one or more desired properties of that system, usually in temporal logic (e.g. CTL or LTL). From the former, a model of the system is constructed, typically a labelled state-transition system in which each state represents a possible system configuration and the transitions represent the evolution of the system from one configuration to another over time. It is then possible to automatically verify whether or not each property is satisfied, based on a systematic and exhaustive exploration of the model.

In probabilistic model checking, the models are augmented with quantitative information regarding the likelihood that transitions occur and the times at which they do so. In practice, these models are typically Markov chains or Markov decision processes. In this chapter, we use *continuous-time Markov chains* (CTMCs), in which transitions between states are assigned (positive, real-valued) rates, which are interpreted as the rates of negative exponential distributions. Properties, while still expressed in temporal logic, are now quantitative in nature. For example, rather than verifying that 'the protein always eventually degrades', we may ask 'what is the probability that the protein eventually degrades?' or 'what is the probability that the protein degrades within t hours?'. The most common temporal logic for this purpose is is CSL (Continuous Stochastic Logic). Furthermore, by adding rewards to a CTMC, we can also specify properties such as 'what is the expected energy dissipation through protein binding within the first t time units?' and 'what is the expected number of binding reactions before relocation occurs?'.

In the remainder of this section we present an introduction to CTMCs, CSL, and the software tool PRISM, which provides support for probabilistic model checking of CTMCs using CSL. For further details, see e.g. [13].

2.1 Continuous-Time Markov Chains

Continuous-time Markov chains (CTMCs), frequently used in performance analysis, model both continuous real time and probabilistic choice. This is done by specifying the *rate* at which a transition between two states occurs. Formally, we have the following definition.

Definition 1. *A CTMC is a tuple $\mathcal{C} = (S, \mathbf{R}, L)$ where: S is a finite set of* states*; $\mathbf{R} : S \times S \to \mathbb{R}_{\geq 0}$ is a* transition rate matrix*; $L : S \to 2^{AP}$ is a* labelling function*.*

The transition rate matrix $\mathbf{R}$ assigns rates to each pair of states, which are used as the parameter of an exponential distribution. A transition can only occur between states s and s' if $\mathbf{R}(s,s')>0$ and, in this case, the probability of this transition being triggered within t time-units equals $1 - e^{-\mathbf{R}(s,s')\cdot t}$. Typically, in a state s, there is more than one state s' for which $\mathbf{R}(s,s')>0$, this is known as a *race condition* and the first transition to be triggered determines the next state. The choice of successor state s' from state s is thus probabilistic. The probability of moving to s' is $\frac{\mathbf{R}(s,s')}{E(s)}$, where $E(s) = \sum_{s' \in S} \mathbf{R}(s,s')$ is the *exit rate* of state s. The total time spent in s before any transition occurs is exponentially distributed with rate $E(s)$. The labelling function L assigns *atomic propositions* from a set AP to each state of the CTMC. These are used to label states with properties of interest.

A CTMC can be augmented with *reward structures* which are used to annotate its states and/or transitions with additional quantitative information. Formally, a reward structure for a CTMC is a pair $(\underline{\rho}, \boldsymbol{\iota})$ where:

- $\underline{\rho} : S \to \mathbb{R}_{\geq 0}$ is the *state reward function*;
- $\boldsymbol{\iota} : S \times S \to \mathbb{R}_{\geq 0}$ is the *transition reward function*.

A path of a CTMC $\mathcal{C} = (S, \mathbf{R}, L)$ is a non-empty sequence $s_0 t_0 s_1 t_1 s_2 \ldots$ where $s_i \in S$, $t_i \in \mathbb{R}_{>0}$ and $\mathbf{R}(s_i, s_{i+1})>0$ for all $i \geq 0$. The value t_i represents the amount of time spent in the state s_i and we denote by $\omega@t$ the state occupied at time t, i.e. s_j where j is the smallest index for which $\sum_{i=0}^{j} t_i \geq t$. We denote by $Path(s)$ the set of all (infinite and finite) paths of the CTMC $\mathcal{C}$ starting in state s. A probability measure Pr_s over $Path(s)$ can then be derived [2].

Two traditional properties of CTMCs are *transient* behaviour, which relates to the state of the model at a particular time instant; and *steady-state* behaviour, which describes the state of the CTMC in the long-run. For a CTMC $\mathcal{C} = (S, \mathbf{R}, L)$, the transient probability $\pi_{s,t}(s')$ is defined as the probability, having started in state s, of being in state s' at time instant t. The steady-state probability $\pi_s(s')$ is the probability of, having started in state s, being in state s' in the long-run. The steady-state probability distribution, i.e. the values $\pi_s(s')$ for all $s' \in S$, can be used to infer the percentage of time, in the long-run, that the CTMC spends in each state.

2.2 Continuous Stochastic Logic

The temporal logic CSL originally introduced by Aziz et al. [1] and since extended by Baier et al. [2] is based on the temporal logics CTL [5] and PCTL [7] and provides a powerful means to specify both path-based and traditional state-based performance measures on CTMCs. We use an extended version [13] which also allows for the specification of reward properties.

Definition 2. *The syntax of CSL is as follows:*

$$\phi ::= \texttt{true} \mid a \mid \neg\phi \mid \phi\wedge\phi \mid \texttt{P}_{\sim p}[\phi\ \texttt{U}^I\ \phi] \mid \texttt{S}_{\sim p}[\phi] \mid$$
$$\texttt{R}_{\sim r}[\texttt{I}^{=t}] \mid \texttt{R}_{\sim r}[\texttt{C}^{\leq t}] \mid \texttt{R}_{\sim r}[\texttt{F}\ \phi] \mid \texttt{R}_{\sim r}[\texttt{S}]$$

where a is an atomic proposition, $\sim\in\{<,\leq,\geq,>\}$, $p\in[0,1]$, I is an interval of $\mathbb{R}_{\geq 0}$ and $r,t\in\mathbb{R}_{\geq 0}$.

CSL formulae are evaluated over the states of a CTMC and we write $s \models \phi$ to indicate the CSL formula ϕ is true is state s. We also say that ϕ *holds* or *is satisfied* in s. CSL includes the standard operators from propositional logic: `true` (satisfied in all states); atomic propositions (a is true in states which are labelled with a); negation ($\neg\phi$ is true if ϕ is not); and conjunction ($\phi_1\wedge\phi_2$ is true if both ϕ_1 and ϕ_2 are true). From these, we can derive other standard Boolean operators such as disjunction ($\phi_1\vee\phi_2\equiv\neg(\neg\phi_1\wedge\neg\phi_2)$) and implication ($\phi_1\Rightarrow\phi_2\equiv\neg\phi_1\vee\phi_2$) in the usual way.

CSL also includes two probabilistic operators, P and S, both of which include a probability bound $\sim p$. A formula $\texttt{P}_{\sim p}[\psi]$ is true in a state s if the probability of the path formula ψ being satisfied from state s meets the bound $\sim p$. In this chapter we use a single type of path formula, $\phi_1\ \texttt{U}^I\ \phi_2$, called an *until* formula, which is true of a path ω if, for some time instant $t\in I$, at time t in the path ω the CSL subformula ϕ_2 is true and the subformula ϕ_1 is true at all preceding time instants. Perhaps the most common use of this operator is the case where ϕ_1 is `true`, in which case $\texttt{P}_{\sim p}[\texttt{true}\ \texttt{U}^I\ \phi]$ states that the probability of ϕ being true at some point in the interval I satisfies $\sim p$. In particular, this allows reasoning about transient behaviour of a CTMC ($\texttt{P}_{\sim p}[\texttt{true}\ \texttt{U}^{[t,t]}\ \phi]$, i.e. the probability of ϕ being true at time instant t) and untimed reachability ($\texttt{P}_{\sim p}[\texttt{true}\ \texttt{U}^{[0,\infty)}\ \phi]$, i.e. the probability of ϕ eventually being true). In the latter case we often omit the interval, i.e. $\texttt{P}_{\sim p}[\texttt{true}\ \texttt{U}\ \phi]\equiv\texttt{P}_{\sim p}[\texttt{true}\ \texttt{U}^{[0,\infty)}\ \phi]$. The S operator is used to specify steady-state behaviour of a CTMC. More precisely, $\texttt{S}_{\sim p}[\phi]$ asserts that the steady-state probability of being in a state satisfying ϕ meets the bound $\sim p$.

CSL also has an R operator for properties concerning the expected value of rewards. The formula $\texttt{R}_{\sim r}[\texttt{I}^{=t}]$ asserts that the expected value of the state reward at time instant t meets the bound $\sim r$. Similarly, $\texttt{R}_{\sim r}[\texttt{C}^{\leq t}]$ refers to the expected reward accumulated up until time t, $\texttt{R}_{\sim r}[\texttt{F}\ \phi]$ to the expected reward accumulated before a state satisfying ϕ is reached, and $\texttt{R}_{\sim r}[\texttt{S}]$ to the long-run average expected reward. Note that the $\texttt{I}^{=t}$ operator refers only to state rewards, whereas the others refer to both state and transition rewards. Also, in the latter case state rewards are interpreted as the rate at which rewards are accumulated, i.e. spending time t in state s accumulates a reward of $\underline{\rho}(s)\cdot t$.

The semantics of CSL over CTMCs is defined as follows.

Definition 3. *Let $\mathcal{C} = (S, \mathbf{R}, L)$ be a labelled CTMC. For any state $s \in S$ the relation $s \models \phi$ is defined inductively by:*

$$
\begin{array}{rcl}
s \models \texttt{true} & & \textit{for all } s \in S \\
s \models a & \Leftrightarrow & a \in L(s) \\
s \models \neg\phi & \Leftrightarrow & s \not\models \phi \\
s \models \phi_1 \wedge \phi_2 & \Leftrightarrow & s \models \phi_1 \wedge s \models \phi_2 \\
s \models \texttt{P}_{\sim p}[\psi] & \Leftrightarrow & \textit{Prob}(s,\psi) \sim p \\
s \models \texttt{S}_{\sim p}[\phi] & \Leftrightarrow & \sum_{s' \models \phi} \pi_s(s') \sim p \\
s \models \texttt{R}_{\sim r}[\texttt{I}^{=t}] & \Leftrightarrow & \textit{Exp}(s, X_{\texttt{I}^{=t}}) \sim r \\
s \models \texttt{R}_{\sim r}[\texttt{C}^{\leq t}] & \Leftrightarrow & \textit{Exp}(s, X_{\texttt{C}^{\leq t}}) \sim r \\
s \models \texttt{R}_{\sim r}[\texttt{F}\ \phi] & \Leftrightarrow & \textit{Exp}(s, X_{\texttt{F}\phi}) \sim r \\
s \models \texttt{R}_{\sim r}[\texttt{S}] & \Leftrightarrow & \lim_{t\to\infty} \frac{1}{t} \cdot \textit{Exp}(s, X_{\texttt{C}^{\leq t}}) \sim r
\end{array}
$$

where $\textit{Prob}(s,\psi) \overset{\text{def}}{=} \textit{Pr}_s\{\omega \in \textit{Path}(s) \,|\, \omega \models \psi\}$, $\textit{Exp}(s, X)$ denotes the expectation of the random variable X with respect to the probability measure $\textit{Pr}_s$ and for any path $\omega = s_0 t_0 s_1 t_1 s_2 \cdots \in \textit{Path}(s)$:

$$
\omega \models \phi_1\ \texttt{U}^I\ \phi_2 \Leftrightarrow \exists t \in I.\,(\,\omega@t \models \phi_2 \wedge \forall x \in [0,t).\,(\omega@x \models \phi_1)\,)
$$

$$
\begin{array}{rcl}
X_{\texttt{I}^{=t}}(\omega) & = & \underline{\rho}(\omega@t) \\
X_{\texttt{C}^{\leq t}}(\omega) & = & \sum_{i=0}^{j_t-1}\big(t_i \cdot \underline{\rho}(s_i) + \boldsymbol{\iota}(s_i, s_{i+1})\big) + \Big(t - \sum_{i=0}^{j_t-1} t_i\Big) \cdot \underline{\rho}(s_{j_t}) \\
X_{\texttt{F}\phi}(\omega) & = & \begin{cases} 0 & \textit{if } \omega(0) \models \phi \\ \infty & \textit{if } \forall i \in \mathbb{N}.\, s_i \not\models \phi \\ \sum_{i=0}^{\min\{j | s_j \models \phi\}-1} t_i \cdot \underline{\rho}(s_i) + \boldsymbol{\iota}(s_i, s_{i+1}) & \textit{otherwise} \end{cases}
\end{array}
$$

and $j_t = \min\{j \mid \sum_{i=0}^{j} t_i \geq t\}$.

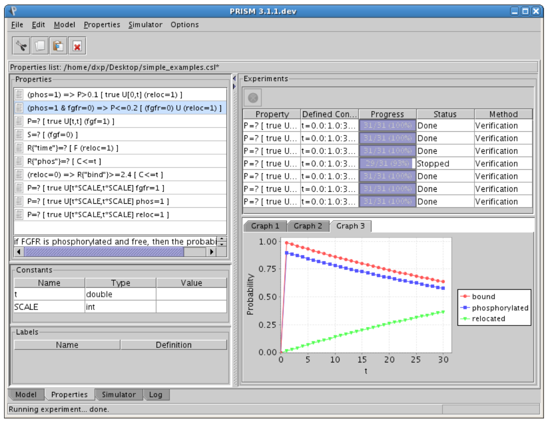
In many cases, it is more useful to generate quantitative, rather than Boolean, results for CSL properties. For this purpose, we allow the bounds $\sim p$ and $\sim r$ attached to the `P`, `S` and `R` operators to be replaced with $=?$. This permits, for example, properties such as:

– $\texttt{P}_{=?}[\texttt{true}\ \texttt{U}^{[t,t]}\ \phi]$ - 'what is the probability of ϕ being true at time t?';
– $\texttt{S}_{=?}[\phi]$ - 'what is the long-run probability of ϕ holding?';
– $\texttt{R}_{=?}[\texttt{F}\ \phi]$ - 'what is the expected reward accumulated before a state satisfying ϕ is reached?'.

The $=?$ form of the `P`, `S` and `R` operators can only be used when it is the outermost operator in a CSL formula.

2.3 PRISM

PRISM [10, 17] is a probabilistic model checker which provides automatic verification of CTMCs using the logic CSL, as described in the previous sections. It also provides support for two other types of probabilistic models: discrete-time Markov chains and Markov decision processes. See e.g. [22] for more information.



Properties list: /home/dxp/Desktop/simple_examples.csl*

```
(phos=1) => P>0.1 [ true U[0,t] (reloc=1) ]
(phos=1 & fgfr=0) => P<=0.2 [ (fgfr=0) U (reloc=1) ]
P=? [ true U[t,t] (fgf=1) ]
S=? [ (fgf=0) ]
R{"time"}=? [ F (reloc=1) ]
R{"phos"}=? [ C<=t ]
(reloc=0) => R{"bind"}>=2.4 [ C<=t ]
P=? [ true U[t*SCALE,t*SCALE] fgfr=1 ]
P=? [ true U[t*SCALE,t*SCALE] phos=1 ]
P=? [ true U[t*SCALE,t*SCALE] reloc=1 ]
```

if FGFR is phosphorylated and free, then the probabi

Constants

Name	Type	Value
t	double	
SCALE	int	

Labels

Name	Definition

Property	Defined Con...	Progress	Status	Method
P=? [true U...	t=0.0:1.0:3...	31/31 (100%)	Done	Verification
P=? [true U...	t=0.0:1.0:3...	31/31 (100%)	Done	Verification
P=? [true U...	t=0.0:1.0:3...	31/31 (100%)	Done	Verification
P=? [true U...	t=0.0:1.0:3...	29/31 (93%)	Stopped	Verification
P=? [true U...	t=0.0:1.0:3...	31/31 (100%)	Done	Verification
P=? [true U...	t=0.0:1.0:3...	31/31 (100%)	Done	Verification
P=? [true U...	t=0.0:1.0:3...	31/31 (100%)	Done	Verification

Running experiment... done.

Fig. 1. A screenshot of the PRISM graphical user interface

The tool accepts probabilistic models described in *the PRISM modelling language*, a simple, state-based language which will be explained in the later sections of this chapter. The basic functionality of PRISM is to take a model described in this formalism, construct the corresponding probabilistic model (in this case, a CTMC) and then perform model checking of one or more CSL properties. PRISM also supports the notion of *experiments*, which is a way of automating multiple instances of model checking. This allows the user to easily obtain the outcome of one or more properties as functions of model and property parameters. The resulting table of values can either be viewed directly, exported for use in another application such as a spreadsheet, or displayed using PRISM's graph plotting tool. This is frequently a good way of identifying interesting patterns or trends in the behaviour of a system. You will see several examples of this throughout the chapter.

Figure 1 shows a screenshot of the PRISM graphical user interface, illustrating the results of a model checking experiment being plotted on a graph. The tool also features a built-in text-editor for the PRISM language and a discrete-event simulation engine which can be used to debug PRISM models. Alternatively, all model checking functionality is also available in a command-line version of the tool. PRISM is a *free*, *open source* application. It presently operates on Linux, Unix, Windows and Macintosh operating systems. Both binary and source code versions can be downloaded from the website [17].

PRISM incorporates a variety of techniques to construct CTMCs and perform CSL model checking on them. Much of the underlying work is a combination of graph-theoretical algorithms, e.g. for reachability analysis, and numerical computation, e.g. to calculate probabilities and expected reward values for each state of the CTMC. For the latter, due to the sizes of the problems typically solved, PRISM uses iterative numerical solution methods. Some aspects of CSL model checking (e.g. those based on long-run properties of a CTMC) require solution of linear equation systems. For these, well-known techniques such as the Jacobi, Gauss-Seidel and SOR (successive over-relaxation) methods are available. For

1: FGF binds/releases FGFR	
FGF + FGFR $\rightarrow$ FGF:FGFR	k_1=5e+8 $M^{-1}s^{-1}$
FGF + FGFR $\leftarrow$ FGF:FGFR	k_2=0.002 s^{-1}
2: Phosphorylation of FGFR (whilst FGF:FGFR)	
FGFR $\rightarrow$ FGFRP	k_3=0.1 s^{-1}
3: Dephosphorylation of FGFR	
FGFRP $\rightarrow$ FGFR	k_4=0.01 s^{-1}
4: Relocation of FGFR (whilst FGFRP)	
FGFR $\rightarrow$ relocFGFR	k_5=1/60 min^{-1}

Fig. 2. Summary of the reactions

timed operators of CSL (i.e. those based on transient CTMC properties), PRISM implements another common iterative numerical method called uniformisation. In addition, the tool provides approximate solution methods based on Monte Carlo techniques and the built-in discrete-event simulation engine.

One final notable aspect of PRISM is that it is a *symbolic* model checker: it is implemented primarily using data structures based on binary decision diagrams (BDDs), for example *multi-terminal* binary decision diagrams (MTBDDs) and other variants. These data structures provide compact representations and efficient manipulation of large, structured probabilistic models by exploiting regularity from the high-level descriptions of the models. In order to maximise efficiency, PRISM actually uses combinations of these symbolic data structures and conventional *explicit* storage schemes such as sparse matrices and arrays. See e.g. [12, 15] for more information about the implementation of PRISM.

3 Modelling a simple biological system in PRISM

We now illustrate how PRISM can be used as a tool for modelling and analysing biological systems. We will do this by developing a case study based on the role of FGF (Fibroblast Growth Factor) in receptor biosynthesis. FGF are a family of proteins which play an important role in cell signalling, e.g., wound healing.

3.1 A simple set of reactions

Figure 2 shows a simple set of reactions based on the role of FGF in receptor biosynthesis, which can be summarised as follows. An FGF ligand (molecule) can bind to an FGF receptor (FGFR) to form the compound FGF:FGFR. Once the compound FGF:FGFR is formed, FGFR can become phosphorylated. These two reactions (binding and phosphorylation) are also reversible. Finally, when FGFR is phosphorylated, it can be relocated (we assume that FGF disappears if it is bound to FGFR when relocation occurs). Each reaction has an associated *kinetic rate* (values k_1 to k_5 in Figure 2). Note that for the binary reaction (binding) the units of the kinetic rate include the concentrations of the reactants, more precisely, the kinetic rate k_1 has units $M^{-1}s^{-1}$, where M refers to the *molar concentration*, i.e. the number of moles per litre.

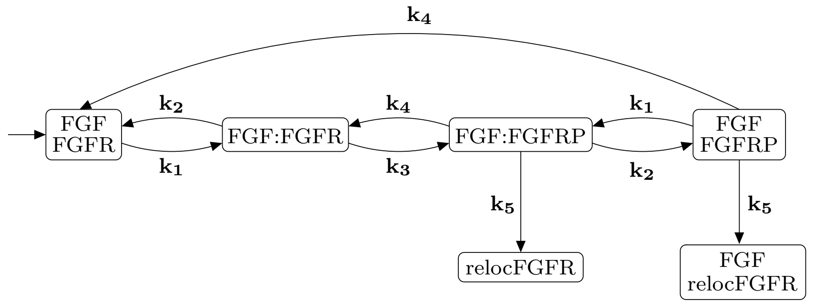


Fig. 3. Underlying CTMC of the system of reactions given in Figure 2

Our approach to modelling the behaviour of biochemical reaction systems is as follows. We consider a collection of molecules, from a variety of molecular species, interacting according to a set of reactions. We make the assumption that the reactions take place in a spatially uniform mixture in a fixed volume at constant pressure and temperature. Under this assumption, two distinct modelling approaches have been proposed. The first, which we refer to as the *continuous deterministic* approach, represents the number of molecules of each species at time t by a continuous function. A set of ordinary differential equations are then derived, the solutions to which give the (average) concentration of each molecular species over time.

The second approach, and the one used in this chapter, is to use a *discrete stochastic* model. In this case, the amount of each molecular species is modelled as a discrete quantity and the occurrence of a reaction between one or more of these molecules is considered a discrete event. The evolution of this model over time is inherently stochastic. In fact, the underlying model can be shown to be a continuous-time Markov chain (CTMC), where the stochastic rates associated with each transition of the CTMC can be derived from the kinetic rates of the reaction system. In the case of unary reactions the stochastic rate equals the kinetic rate. For binary reactions, supposing the kinetic rate is given in terms of molar concentrations, the stochastic rate is obtained by dividing by $V \cdot \mathcal{N}$ where V is the volume and $\mathcal{N}$ is Avogadro's number. For a more in-depth coverage of this topic, see for example [24, 6].

We now return to the simple example described above. The different possible species which can be present in the system are as follows: free FGF (FGF), free FGFR (FGFR), FGF bound to FGFR (FGF:FGFR), FGF bound to phosphorylated FGFR (FGF:FGFRP), free phosphorylated FGFR (FGFRP) and relocated FGFR (relocFGFR). Making the assumption that the system initially comprises a single FGF ligand and a single FGFR receptor, the corresponding CTMC for this example is given in Figure 3. Where the stochastic rates ($\mathbf{k_i}$) are derived from the kinetic rates (k_i) via the calculation described in the previous paragraph.

```
ctmc

const double k1 = 5000; // rate of binding
const double k2 = 0.002; //rate of release
const double k3 = 0.1; // rate of phosphorylation
const double k4 = 0.01; // rate of dephosphorylation
const double k5 = 1/(60 * 60); // rate of relocation

module FGF

    fgf : [0..2] init 0; // 0 - free, 1 - bound, 2 - removed from system

    [bind] fgf=0 → (fgf'=1); // FGF and FGFR bind
    [rel]  fgf=1 → (fgf'=0); // FGF and FGFR unbind
    [reloc] fgf=1 → (fgf'=2); // FGF disappears since bound when FGFR relocates

endmodule

module FGFR

    fgfr  : [0..1] init 0; // 0 - free, 1 - bound
    phos  : [0..1] init 0; // 0 - unphosphorylated, 1 - phosphorylated
    reloc : [0..1] init 0; // 0 - not relocated, 1 - relocated

    [bind] fgfr=0 & reloc=0 → (fgfr'=1); // FGF and FGFR bind
    [rel] fgfr=1 & reloc=0 → (fgfr'=0); // FGF and FGFR unbind
    [] fgfr=1 & phos=0 & reloc=0 → k3 : (phos'=1); // FGFR phosphorylates
    [] phos=1 & reloc=0 → k4 : (phos'=0); // FGFR dephosphorylates
    [] phos=1 & fgfr=0 & reloc=0 → k5 : (reloc'=1) & (fgfr'=0) & (phos'=0); // Relocates
    [reloc] phos=1 & fgfr=1 & reloc=0 → (reloc'=1) & (fgfr'=0) & (phos'=0); // Relocates

endmodule

module RATES

    [bind] true → k1 : true; // FGF and FGFR bind
    [rel] true → k2 : true; // FGF and FGFR unbind
    [reloc] true → k5 : true; // FGFR relocates

endmodule
```

Fig. 4. First possible PRISM representation

3.2 PRISM models

We now give two alternative approaches for modelling these reactions in PRISM, shown in Figures 4 and 5, respectively. A model described in the PRISM language comprises a keyword, corresponding to the model type (`ctmc` in this case), and a set of *modules*, the state of each being represented by a set of finite-ranging *variables*. The behaviour of this module, i.e. the changes in states which it can undergo, is specified by a number of *guarded commands* of the form $[]\ g \rightarrow r : u$. The interpretation of a command is that if the predicate (guard) g is true, then the system is updated according to u, which comprises one or more statements of the form $(x' = \dots)$ indicating how the value of variable x is changed. The rate at which this occurs is given by r, i.e. this is the value that will be attached to the corresponding transition in the underlying CTMC.

PRISM supports *synchronisation* between modules in the style of process algebras. This is achieved by labelling commands with *actions* (placed between the initial square brackets). Transitions in different modules labelled with the

```
ctmc

const double k1 = 5000; // rate of binding
const double k2 = 0.002; //rate of release
const double k3 = 0.1; // rate of phosphorylation
const double k4 = 0.01; // rate of dephosphorylation
const double k5 = 1/(60 * 60); // rate of relocation

module SYSTEM

    x : [0..5] init 0;
    // 0 - FGF and FGFR free
    // 1 - FGF and FGFR free (FGFR phosphorylated)
    // 2 - FGF and FGFR bound
    // 3 - FGF and FGFR bound (FGFR phosphorylated)
    // 4 - FGF free and FGFR relocated
    // 5 - FGFR relocated

    // FGF and FGFR bind
    [] x=0 → k1 : (x'=2); // FGFR not phosphorylated
    [] x=1 → k1 : (x'=3); // FGFR phosphorylated
    // FGF and FGFR unbind
    [] x=2 → k2 : (x'=0); // FGFR not phosphorylated
    [] x=3 → k2 : (x'=1); // FGFR phosphorylated
    // FGFR becomes phosphorylated (FGF must be bound)
    [] x=2 → k3 : (x'=3);
    // FGFR dephosphorylates
    [] x=1 → k4 : (x'=0); // FGF not bound
    [] x=3 → k4 : (x'=2); // FGF bound
    // FGFR relocates (FGFR must be phosphorylated)
    [] x=1 → k5 : (x'=4); // FGF not bound
    [] x=3 → k5 : (x'=5); // FGF bound

endmodule
```

Fig. 5. Second possible PRISM representation

same action occur simultaneously. The rate of synchronised transitions is equal to the product of the individual rates of the commands of the different modules that synchronise. Since the product of several rates is not always meaningful, a common technique, as seen here, is to make one action *active*, which actually defines the rate for the synchronised transition, and the others *passive* with rate 1. In PRISM, when a rate is omitted for a command it is assumed to be 1. By default, all modules are combined using the standard CSP parallel composition [21] (i.e. modules synchronise over all their common actions). In addition, by using the `system` ... `endsystem` construct, several other CSP operators, including asynchronous parallel composition and hiding, can be employed when combining PRISM modules. Some of these constructs will be illustrated in later examples. For further details see the PRISM manual [17].

In our first approach (Figure 4), we represent the two main molecules (FGF and FGFR) as separate modules, each with variables representing their current state (details are given in the comments after each variable declaration). Reactions involving more than one species are modelled using synchronisation. Consider for example the binding of FGF and FGFR, modelled by the first commands of modules *FGF* and *FGFR* and synchronisation on the action label *bind*. In this case, the variables *fgf* and *fgfr* in the two modules simultaneously change from 0 to 1. For this model, we use a third module *RATES* to store the rates of

```
// time spent in any state
rewards "time"

    true : 1;

endrewards
```

```
// time spent phosphorylated
rewards "phos"

    phos=1 : 1;

endrewards
```

```
// number of bindings
rewards "bind"

    [bind] true : 1;

endrewards
```

Fig. 6. Three reward structures for the PRISM model of Figure 4

synchronous transitions. This module also takes place in the synchronisation on *bind* and the overall rate is thus *k1*. Reactions involving only a single protein can be modelled without synchronisation, as seen for example in the third, fourth and fifth commands of module *FGFR*.

In our second approach (Figure 5), we use a single module with one variable x representing the (six) possible states of the whole system. The meaning of each value of x is given in the comments and these correspond directly to the states of the CTMC shown in Figure 3. These two PRISM models result in identical CTMCs. The second is a more concise description in PRISM but the first has a more intuitive state encoding and is hence easier to modify and to express properties for. In general, a combination of the above two modelling approaches is used: in simple cases it is possible to use a single variable, but as the system becomes more complex the use of separate variables and synchronisation becomes more desirable. We will see this as we develop this case study further in Sections 4 and 5.

Finally, we add reward structures to our model. In PRISM, these are described using the

`rewards` "*reward_name*" ... `endrewards`

construct and are specified using multiple reward items of the form

$$g \ : \ r; \quad \text{or} \quad [a]\, g \ : \ r;$$

to describe state and transition rewards, respectively. In the above, g is a predicate (over all the variables of the model), a is an action label appearing in the commands of the model and r is a real-valued expression (which can contain any variables, constants, etc. from the model). A reward item "$g \ : \ r$" assigns a state reward of r to all states satisfying g and a reward item "$[a] \quad g \ : \ r$" assigns a transition reward of r to all a-labelled transitions from states satisfying g. Multiple rewards (from different reward items) for a single state or transition are summed and states or transitions with no assigned reward are assumed to have reward 0.

Figure 6 shows three such reward structures for the PRISM model of Figure 4. The first reward structure ("*time*") simply assigns a state reward of 1 to all states in the model. This can be used, for example, to analyse the total expected time before some event occurs. The second reward structure ("*phos*") assigns a state reward of 1 only to states in which FGFR is phosphorylated. This could be used to compute the amount of time which FGFR spends phosphorylated within a particular period of time or the expected amount of phosphorylated FGFR at a specific time instant. Lastly, the reward structure "*bind*" assigns a reward of 1 to all transitions which correspond to a binding between FGF and FGFR.

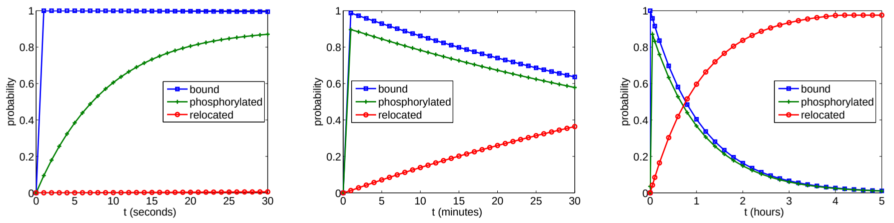
3.3 Model analysis with PRISM

We now demonstrate how the temporal logic CSL (see Section 2.2) can be used to express properties of the reaction system model presented in Figure 4, augmented with the reward structures from the previous section. Below are some CSL examples together with their informal meaning. Since there are multiple reward structures, we will adopt PRISM's notation $\mathtt{R}\{\text{"}rew\text{"}\}$ to indicate use of the CSL reward operator $\mathtt{R}$ using the reward structure with name "*rew*". Note also that, in our model, the unit of time used is seconds.

- $(phos{=}1) \Rightarrow \mathtt{P}_{>0.1}[\ \mathtt{true}\ \mathtt{U}^{[0,t]}\ (reloc{=}1)\ ]$ - 'if FGFR is currently phosphorylated, then the probability of it being relocated within the next t seconds is greater than 0.1';
- $(phos{=}1 \wedge fgfr{=}0) \Rightarrow \mathtt{P}_{\leq 0.2}[\ (fgfr{=}0)\ \mathtt{U}\ (reloc{=}1)\ ]$ - 'if FGFR is phosphorylated and free, then the probability of it being relocated before binding to FGF is at most 0.2';
- $\mathtt{P}_{=?}[\ \mathtt{true}\ \mathtt{U}^{[t,t]}\ (fgf{=}1)\ ]$ - 'the probability that FGF is bound to FGFR at time instant t (i.e. after exactly t seconds)';
- $\mathtt{P}_{=?}[\ \mathtt{true}\ \mathtt{U}\ (reloc{=}1 \wedge fgf{=}2)\ ]$ - 'the probability that FGFR relocates and FGF is bound when relocation occurs';
- $\mathtt{S}_{=?}[\ (fgf{=}0)\ ]$ - 'the probability that, in the long run, FGF is free';
- $\mathtt{R}\{\text{"}time\text{"}\}_{=?}[\ \mathtt{F}\ (reloc{=}1)\ ]$ - 'the expected time taken before FGFR relocates'
- $\mathtt{R}\{\text{"}phos\text{"}\}_{=?}[\ \mathtt{C}^{\leq t}\ ]$ - 'the expected time that FGFR spends phosphorylated within the next t seconds';
- $(reloc{=}0) \Rightarrow \mathtt{R}\{\text{"}bind\text{"}\}_{\geq 2.4}[\ \mathtt{C}^{\leq t}\ ]$ - 'if FGFR is not relocated, the expected number of bindings during the next t seconds is at least 2.4';

When analysing quantitative properties of system such as these, it is often also useful to study how the properties vary as changes are made to parameters either in the properties (e.g. t above) or in the model. Analysis of this kind is much more likely to provide insight into the model or to identify interesting or anomalous behaviour.

To illustrate this, Figure 7 shows results obtained with PRISM for the probabilities that, at time instant t, FGFR is: (i) bound to FGF; (ii) phosphorylated; (iii) relocated. The first of these, for example, uses the CSL property $\mathtt{P}_{=?}[\ \mathtt{true}\ \mathtt{U}^{[t,t]}\ (fgf{=}1)\ ]$ from the list above. Results are plotted for ranges of t over three different time scales (seconds, minutes and hours). Figure 7(a) shows that in the initial evolution of the system FGF and FGFR bind very quickly (and remain bound) after which FGFR becomes phosphorylated while there is almost no chance of FGFR relocating. Figure 7(b) and Figure 7(c) show however that, as time elapses, the chance that FGF and FGFR are bound diminishes and the chance that FGFR is phosphorylated diminishes faster. In addition we see that eventually FGFR will become relocated.



(a) Time scale seconds (b) Time scale minutes (c) Time scale hours

Fig. 7. Transient properties of FGFR for the model of Figure 4

3.4 Exercises

1. Based on the model in Figure 4 and the reward structures in Figure 6, write CSL specifications for the following properties:
   (a) 'if FGFR is currently phosphorylated, then the probability that it remains phosphorylated until relocation occurs is at most 0.65';
   (b) 'the probability that FGFR is phosphorylated at time instant t';
   (c) 'the probability that FGFR is phosphorylated for the first time within the time interval $[t_1, t_2]$';
   (d) 'the expected time that FGFR is phosphorylated before it relocates';
   (e) 'the expected number of times that FGF and FGFR bind before FGFR relocates'.
2. Construct appropriate reward structures for properties of the model in Figure 4 relating to the expected time that FGF and FGFR are bound and the expected number of bindings *and* unbindings. Write CSL specifications for calculating the expected time that FGF and FGFR spend bound during the first t seconds, the expected number of bindings and unbindings in this time, and the expected time spent bound before relocation occurs.
3. Extend the model of Figure 4 with a variable to count the number of phosphorylations and write CSL specifications for the properties:
   (a) 'the probability that at least l phosphorylations occur within the first t seconds';
   (b) 'the probability that at most l phosphorylations occur before relocation'.

   Hint: Since the variable you add to the model must be bounded, make sure that the bound is larger than required for the property.
4. Based on the model of Figure 4, write a new version in which each of the six possible species (FGF, FGFR, FGF:FGFR, FGF:FGFRP, FGFRP, relocFGFR) is represented by a separate PRISM module. Check that the states and transitions in the new model match those of the original one and that numerical results such as those in Figure 7 agree.
   Hint: In this model the relevant modules will need to synchronise when phosphorylation and dephosphorylation occurs.

```
ctmc

const double N = 3; // number of each type

const double k1 = 5000/N; // rate of binding
const double k2 = 0.002; //rate of release
const double k3 = 0.1; // rate of phosphorylation
const double k4 = 0.01; // rate of dephosphorylation
const double k5 = 1/(60 * 60); // rate of relocation

module FGF .... endmodule

// construct further FGF molecules through renaming
module FGF2 = FGF [ fgf=fgf2 ] endmodule
module FGF3 = FGF [ fgf=fgf3 ] endmodule

module FGFR .... endmodule

// construct further FGF molecules through renaming
module FGFR2 = FGFR [ fgfr=fgfr2, phos=phos2, reloc=reloc2 ] endmodule
module FGFR3 = FGFR [ fgfr=fgfr3, phos=phos3, reloc=reloc3 ] endmodule

module RATES

    [bind] true → k1 : true; // FGF and FGFR bind
    [rel] fgf + fgf2 + fgf3>0 → k2/(fgf + fgf2 + fgf3) : true; // FGF and FGFR unbind
    [reloc] fgf + fgf2 + fgf3>0 → k2/(fgf + fgf2 + fgf3) : true; // FGFR relocates

endmodule

system // system definition (molecules of the same type do not interact)

    (FGF ||| FGF2 ||| FGF3) || (FGFR ||| FGFR2 ||| FGFR3) || RATES

endsystem
```

Fig. 8. Individual-based PRISM representation (extending Figure 4)

4 Modelling larger numbers of molecules

4.1 Individual-based representations

In this section we extend our existing model by allowing more than one of each FGF and FGFR molecule (and thus of the other various species) to occur in the system. One possible approach to doing this is to model each individual molecule in the system separately. This is sometimes necessary when the properties of interest for the system refer to the behaviour of a particular molecule.

Figure 8 shows PRISM code that can be used to extend the previous model of Figure 4 to contain three of each molecule. First, we add a new PRISM module for each extra molecule. This is done using a PRISM language feature called *renaming*: this creates a copy of an existing module, identical except for its variable names which are given new names (in fact, other constants and actions can also be renamed in the same way). This is equivalent to writing out each new module separately but is less error-prone and improves the readability and scalability of the model.

Figure 8 also illustrates the use of PRISM's `system ... endsystem` construct which describes how the modules in the system are composed to produce the full model. Since the FGF modules do not interact with each other, they are composed using the asynchronous parallel operator (|||), which stops them from synchronising on any actions. The same is true for the FGFR modules. The (standard) parallel operator (||) is then used to compose these sub-systems, i.e. we make these sub-systems synchronise over their common actions. If the asynchronous parallel operator had not been used here, we would have needed to introduce a different action label for each possible interactions between FGF and FGFR (e.g. actions of the form *bind_i_j* for binding of the ith FGF molecule with the jth FGFR molecule) to stop FGF (or FGFR) molecules reacting amongst themselves. PRISM also supports several other CSP-based process-algebraic operators to aid modelling of complex systems; see the manual [17] for details.

We also need to modify the reaction rates of the system when increasing the number of molecules. More precisely, supposing that the volume of the system remains proportional to the initial number of FGF molecules, then the rates of binary reactions in Figure 8 (i.e. k_1) are obtained from those in Figure 4 by dividing by the initial number of FGF molecules (see the earlier discussion of computing stochastic reaction rates in Section 3.1).

4.2 Population-based representations

As might be expected, a potential problem with the model outlined above is scalability. The individual-based modelling approach will suffer from the well known state-space explosion problem where, as the complexity of the system under study increases, there is an exponential growth in the state space of the underlying model.

An alternative is to employ a 'population'-based approach where the number of each type of molecule or species is modelled, rather than the state of each individual component. In terms of the PRISM modelling language, this is achieved by using variables as counters, i.e. there is a counter for each of the possible species that can be present in the system (in this case free FGF, free FGFR, free phosphorylated FGFR, FGF bound to FGFR, FGF bound to phosphorylated FGFR and relocated free FGFR). In such a model, for example, an FGF ligand can bind with a FGFR receptor if the counters for the number of free FGF ligands and FGFR receptors are both greater than 0, and the occurrence of such a reaction is modelled by decrementing these counters and incrementing the counter representing the number of FGF ligands bound to FGFR receptors.

In Figure 9 we give a population-based PRISM model for the running example. This is based on the earlier model representing a single instance of each species (Figure 5). Note that we must adjust the rates of the CTMC to take into account the different possible interactions encoded. For example, if there are three FGF ligands (FGF_1, FGF_2 and FGF_3) and two FGFR receptors ($FGFR_1$ and $FGFR_2$) then there are six different possible species: FGF_1:$FGFR_1$, FGF_1:$FGFR_2$, FGF_2:$FGFR_1$, FGF_2:$FGFR_2$, FGF_3:$FGFR_1$ and FGF_3:$FGFR_2$. This is achieved by multiplying the rate of interaction by the number of species of each type that can take part in the reaction and, as can be seen in Figure 9, since rates in the PRISM language can be expressions involving variables, this is straightforward to achieve in the PRISM modelling language.

```
ctmc

const int N; // number of possible elements of FGF
const int M; // number of possible elements of FGFR
const int K = min(N, M); // number of possible elements of FGF:FGFR

const double k1 = 5000/N; // rate of binding
const double k2 = 0.002; //rate of release
const double k3 = 0.1; // rate of phosphorylation
const double k4 = 0.01; // rate of dephosphorylation
const double k5 = 1/(60 * 60); // rate of relocation

module POPULATION_MODEL

    fgf : [0..N] init N; // free FGF
    fgfr : [0..M] init M; // free FGFR (not phosphorylated)
    fgfrp : [0..M] init 0; // free FGFR (phosphorylated)
    bnd : [0..K] init 0; // bound FGF:FGFR (FGFR not phosphorylated)
    bndp : [0..K] init 0; // bound FGF:FGFR (FGFR phosphorylated)
    reloc : [0..M] init 0; // relocated FGFR

    // FGF and FGFR bind
    [] fgf>0 & fgfr>0 & bnd<K
        → fgf * fgfr * k1  : (fgf'=fgf − 1) & (fgfr'=fgfr − 1) & (bnd'=bnd + 1);
    [] fgf>0 & fgfrp>0 & bndp<K
        → fgf * fgfrp * k1 : (fgf'=fgf − 1) & (fgfrp'=fgfrp − 1) & (bndp'=bndp + 1);
    // FGF and FGFR unbind
    [] fgf<N & fgfr<M & bnd>0
        → bnd * k2  : (fgf'=fgf + 1) & (fgfr'=fgfr + 1) & (bnd'=bnd − 1);
    [] fgf<N & fgfrp<M & bndp>0
        → bndp * k2 : (fgf'=fgf + 1) & (fgfrp'=fgfrp + 1) & (bndp'=bndp − 1);
    // FGFR becomes phosphorylated (FGF must be bound)
    [] bnd>0 & bndp<K → bnd * k3 : (bnd'=bnd − 1) & (bndp'=bndp + 1);
    // FGFR dephosphorylates
    [] fgfrp>0 & fgfr<M → fgfrp * k4 : (fgfrp'=fgfrp − 1) & (fgfr'=fgfr + 1);
    [] bndp>0 & bnd<K   → bndp * k4  : (bndp'=bndp − 1) & (bnd'=bnd + 1);
    // FGFR relocates (FGFR must be phosphorylated)
    [] fgfrp>0 & reloc<M → fgfrp * k5 : (fgfrp'=fgfrp − 1) & (reloc'=reloc + 1);
    [] bndp>0 & reloc<M  → bndp * k5  : (bndp'=bndp − 1) & (reloc'=reloc + 1);

endmodule
```

Fig. 9. Population-based PRISM model (extending Figure 5)

```
// amount of phosphorylated FGFR
rewards "phos"

    true : fgfrp + bndp;

endrewards
```

```
// number of bindings
rewards "bind"

    [bind] true : 1;

endrewards
```

Fig. 10. Reward structures for the PRISM model given in Figure 9

Table 1 lists the number of states and transitions in the individual- and population-based models as the initial number of FGF ligands and FGFR receptors ranges between 1 and 10. As can be seen there is a rapid increase in the number of states for the individual-based approach, quickly making model checking infeasible. On the other hand, the results for the population-based model show a far more gradual increase in the state space, allowing analysis of much larger models.

(N,M)	Individual-based		Population-based	
	States	Transitions	States	Transitions
(1,1)	6	9	6	9
(2,2)	52	190	21	57
(3,3)	492	3185	56	193
(4,4)	4,816	48,804	126	509
(5,5)	47,916	699,407	252	1,140
(6,6)	480,880	9,548,278	462	2,275
(7,7)	4,849,620	125,662,133	792	4,166
(8,8)	49,045,120	1,606,974,376	1,287	7,137
(9,9)	496,798,620	20,079,205,667	2,002	11,593
(10,10)	5,036,699,152	246,135,326,874	3,003	18,029

Table 1. States and transitions for individual and population based models

4.3 Model analysis with PRISM

Two reward structures for the population model are presented in Figure 10. The first reward structure ("*phos*") assigns a reward to each state equal to the sum of the variables *fgfrp* and *bndp*, i.e. a reward equal to the amount of FGFR that is phosphorylated. Note the difference between "*phos*" in Figure 10 and Figure 6: instead of assigning states with reward 1 or 0 depending on whether FGFR is phosphorylated or not, we now assign a reward equal to the sum of the two variables indicating the amount of each of the two phosphorylated forms of FGFR.

The second reward structure ("*bind*") associates a reward of 1 with all transitions of the CTMC corresponding to the binding of FGF and FGFR. To do this, we assume that we have added the label *bind* to the commands corresponding to FGF and FGFR binding (i.e. to the first and second commands of the module *POPULATION_MODEL* given in Figure 9). This example demonstrates that action labels can be used purely to specify reward structures, with no influence on the interaction between modules (it is of course important to ensure that additional unwanted interactions are not introduced in this way, i.e. by modules synchronising when they should not).

We now use the temporal logic CSL and reward structures given above to express a number of properties of the population-based model given in Figure 9.

- $\mathtt{P}_{=?}[\ \mathtt{true}\ \mathtt{U}\ (\mathit{reloc}{\geq}l_1 \wedge \mathit{fgf}{\leq}l_2)\ ]$ - 'the probability of reaching a situation where there are at least l_1 relocated FGFR receptors and the number of free FGF ligands is at most l_2';
- $(\mathit{bnd}{>}l_1) \Rightarrow \mathtt{P}_{\geq 0.7}[\ \mathtt{true}\ \mathtt{U}^{[0,t]}\ (\mathit{reloc}{\geq}l_2)\ ]$ - 'if the number of FGF:FGFR compounds is greater than l_1, then the probability that the amount of relocated FGFR will reach l_2 within the next t seconds is at least 0.7';
- $\mathtt{P}_{=?}[\ (\mathit{reloc}{=}0)\ \mathtt{U}\ (\mathit{fgfp}{=}l)\ ]$ - 'the probability that l FGFR receptors are phosphorylated before any FGFR is relocated';
- $(\mathit{fgfr}{=}M) \Rightarrow \mathtt{P}_{<0.05}[\ (\mathit{fgfr}{\geq}l_1)\ \mathtt{U}^{[t_1,t_1+t_2]}\ (\mathit{fgfr}{<}l_2)\ ]$ - 'if all FGFR receptors are free and unphosphorylated, then the probability that at least l_1 FGFR receptors remain free and unphosphorylated until time instant t_1 and within the next t_2 seconds this number drops below l_2 is less than 0.05';

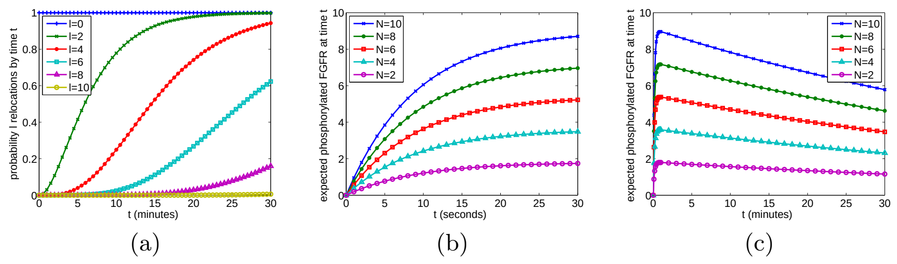


Fig. 11. Transient properties for the population example: (a) probability of l relocations by time t; (b/c) expected phosphorylated FGFR at time t (seconds/minutes).

- $(fgfp + bndp{=}l) \Rightarrow \mathtt{R}\{\text{“}phos\text{”}\}_{\geq 6.8}[\ \mathtt{I}^{=t}\ ]$ - ‘if l FGFR receptors are phosphorylated, the expected number of FGFR receptors phosphorylated t seconds later is at least 6.8’;
- $\mathtt{R}\{\text{“}bind\text{”}\}_{=?}[\ \mathtt{F}\ reloc{=}l\ ]$ - ‘the expected number of times that an FGF ligand binds with an FGFR receptor before l FGFR receptors have relocated’.

As in the previous section, we also give an illustration of the kind of quantitative results than can be obtained. Figure 11(a) shows the probability of l relocations by time t for a range of different values of l when there are initially both 10 FGF ligands and 10 FGFR receptors. Figure 11(b)–(c) shows the expected amount of phosphorylated FGFR at time t for a variety of initial configurations and two different time scales (seconds and minutes). The latter graphs demonstrate that after a rapid increase in phosphorylated FGFR there is a gradual decrease as the amount of relocated FGFR increases and starting from a larger concentration of FGFR and FGF leads to an increase in the amount of phosphorylated FGFR.

4.4 Exercises

1. Based on the model in Figure 9 and the reward structures in Figure 10, write CSL specifications for the following properties:
   (a) ‘the probability that the number of free FGF ligands does not drop below l_1 until after l_2 FGFR receptors have been relocated’;
   (b) ‘if the number of phosphorylated FGFR receptors is at least l, then the probability that all FGFR receptors have been relocated by time t is greater than 0.6’;
   (c) ‘the probability that no FGF ligands are bound to FGFR receptors until time t_1 and, before another t_2 seconds pass, a binding occurs’;
   (d) ‘if at most l FGFR receptors are bound, then the expected number of bindings that occur within t seconds is at least 12.4’.
2. Construct appropriate reward structures for properties of the model in Figure 9 relating to the expected time that at least l FGFR receptors are phosphorylated and the expected number of phosphorylations.
   Hint: You may first need to change the PRISM model description for this.
   Using these reward structures, write specifications for calculating:

(a) 'the expected time that at least l FGFR receptors are phosphorylated up until time t';
(b) 'the expected time that at least l FGFR receptors are phosphorylated before $M{-}l$ receptors have been relocated';
(c) 'the expected number of phosphorylations before a relocation occurs';
(d) 'the expected number of phosphorylations by time t'.

3. Extend the model in Figure 9 with variables to count the number of times that an FGF ligand binds and the number of times that an FGF ligand unbinds with an FGFR receptor and write CSL specifications for the properties:
   (a) 'the probability that at most l reactions between FGF ligands and FGFR receptors (i.e. bindings or unbindings) occur within the first t seconds';
   (b) 'if no FGF ligands have bound to FGFR receptors, then the probability that l bindings occur before any of unbindings occur is at least 0.5'.

5 A more complex model: Compartments

In this section we extend the population model of the previous section to include behaviour relating to the relocated FGFR. The model is motivated by the fact that FGF ligands can only interact with FGFR receptors at the cell surface; relocation causes FGFR to move inside the cell where it loses sight of the ligand. Inside the cell we suppose that either the FGFR is *recycled* back to the surface or it is destroyed by *degradation*. To incorporate this behaviour into our model we divide the system into the following compartments:

main compartment: in this compartment FGF ligands interact with FGFR receptors (i.e. at the cell surface);
recycling compartment: this compartment receives FGFR receptors from the main compartment and at a rate of k_6 (=1/30 min^{-1}) re-introduces the FGFR receptors back into the main compartment;
degradation compartment: this compartment also receives FGFR receptors from the main compartment, but in this case the proteins are degraded at a rate of k_7 (=1/15 min^{-1}).

In this model, we keep the assumption that the ligand FGF disappears if it is bound to FGFR when relocation occurs. In addition, we suppose that the choice between whether the proteins reach either the recycling or degradation compartment from the main compartment is probabilistic: the recycling compartment is reached with probability p and the degradation compartment is reached with probability $1{-}p$. Considering the rates in the CTMC representation of this model it follows that if r is the rate in the CTMC that FGFR receptors leave the main compartment, then the rate of FGFR being relocated to the recycling compartment is $p{\cdot}r$ and the rate of being relocated to the degradation compartment is $(1{-}p){\cdot}r$.

```
....

const double k6 = 1/(30 ∗ 60); // half an hour to return
const double k7 = 1/(15 ∗ 60); // time to degrade
const double p; // probability relocation causes recycling as opposed to degradation

module MAIN_COMPARTMENT
    fgf    : [0..N] init N; // free FGF
    fgfr   : [0..M] init M; // free FGFR (not phosphorylated)
    fgfrp  : [0..M] init 0; // free FGFR (phosphorylated)
    bnd    : [0..M] init 0;  // bound FGF:FGFR (FGFR not phosphorylated)
    bndp   : [0..M] init 0;  // bound FGF:FGFR (FGFR phosphorylated)
    // FGF and FGFR bind
    [] fgf>0 & fgfr>0 & bnd<K
        → fgf ∗ fgfr ∗ k1 : (fgf′=fgf − 1) & (fgfr′=fgfr − 1)   & (bnd′=bnd + 1);
    [] fgf>0 & fgfrp>0 & bndp<K
        → fgf ∗ fgfrp ∗ k1 : (fgf′=fgf − 1) & (fgfrp′=fgfrp − 1) & (bndp′=bndp + 1);
    // FGF and FGFR unbind
    [] fgf<N & fgfr<M & bnd>0
        → bnd ∗ k2 : (fgf′=fgf + 1) & (fgfr′=fgfr + 1)   & (bnd′=bnd − 1);
    [] fgf<N & fgfrp<M & bndp>0
        → bndp ∗ k2 : (fgf′=fgf + 1) & (fgfrp′=fgfrp + 1) & (bndp′=bndp − 1);
    // FGFR becomes phosphorylated
    [] bnd>0 & bndp<K → bnd ∗ k3 : (bnd′=bnd − 1) & (bndp′=bndp + 1);
    // FGFR becomes dephosphorylated
    [] fgfr<M & fgfrp>0 → fgfrp ∗ k4 : (fgfr′=fgfr + 1) & (fgfrp′=fgfrp − 1);
    [] bnd<K & bndp>0    → bndp ∗ k4 : (bnd′=bnd + 1) & (bndp′=bndp − 1);
    // phosphorylated FGFR leave main compartment
    [reloc2] bndp>0 → bndp ∗ p ∗ k5        : (bndp′=bndp − 1);
    [reloc3] bndp>0 → bndp ∗ (1 − p) ∗ k5  : (bndp′=bndp − 1);
    [reloc2] fgfrp>0 → fgfrp ∗ p ∗ k5       : (fgfrp′=fgfrp − 1);
    [reloc3] fgfrp>0 → fgfrp ∗ (1 − p) ∗ k5 : (fgfrp′=fgfrp − 1);
    // FGFR arrives from compartment 2
    [recyc]  fgfr<M → (fgfr′=fgfr + 1);
    [recycp] fgfrp<M → (fgfrp′=fgfrp + 1);
endmodule

module RECYCLING_COMPARTMENT
    fgfr2  : [0..M] init 0; // free FGFR (not phosphorylated)
    fgfrp2 : [0..M] init 0; // free FGFR (phosphorylated)
    // FGFR relocates
    [reloc2] fgfrp2<M → (fgfrp2′=fgfrp2 + 1);
    // FGFR dephosphorylates
    [] fgfr2<M & fgfrp2>0 → fgfrp2 ∗ k4 : (fgfr2′=fgfr2 + 1) & (fgfrp2′=fgfrp2 − 1);
    // FGFR returns to main compartment
    [recyc]  fgfr2>0  → fgfr2 ∗ k6 : (fgfr2′=fgfr2 − 1);
    [recycp] fgfrp2>0 → fgfrp2 ∗ k6 : (fgfrp2′=fgfrp2 − 1);
endmodule

module DEGRADATION_COMPARTMENT
    fgfr3  : [0..M] init 0; // free FGFR (not phosphorylated)
    fgfrp3 : [0..M] init 0; // free FGFR (phosphorylated)
    deg3   : [0..M] init 0; // degraded FGFR (not phosphorylated)
    degp3  : [0..M] init 0; // degraded FGFR ( phosphorylated)
    // FGFR relocates
    [reloc3] fgfrp3<M → (fgfrp3′=fgfrp3 + 1);
    // FGFR dephosphorylates
    [] fgfr3<M & fgfrp3>0 → fgfrp3 ∗ k4 : (fgfr3′=fgfr3 + 1) & (fgfrp3′=fgfrp3 − 1);
    // FGFR degrades
    [] deg3<M & fgfr3>0  → fgfr3 ∗ k7 : (deg3′=deg3 + 1) & (fgfr3′=fgfr3 − 1);
    [] degp3<M & fgfrp3>0 → fgfrp3 ∗ k7 : (degp3′=degp3 + 1) & (fgfrp3′=fgfrp3 − 1);
endmodule
```

Fig. 12. Compartments PRISM model (extending Figure 9)

```
// amount of phosphorylated FGFR
rewards "phos"

    true : fgfrp + bndp;
    true : fgfrp2;
    true : fgfrp3;

endrewards
```

```
// number of relocations
rewards "reloc"

    [reloc2] true : 1;
    [reloc3] true : 1;

endrewards
```

Fig. 13. Reward structures for the PRISM model of Figure 12

5.1 The PRISM model

Figure 12 shows a PRISM language model for this system based on the population model of the previous section (see Figure 9). Each compartment is modelled by a separate module and transitions between compartments (relocations and re-introductions) are modelled through the modules synchronising on actions.

Figure 13 gives two reward structures for this model. The first reward structure ("*phos*") assigns a reward to each state equal to the total number of phosphorylated FGFR receptors in the entire system (i.e. in any of the compartments). Note that we could also have expressed this reward in a single line as:

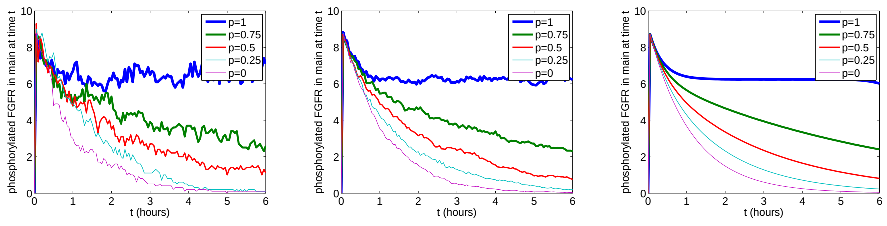
$$\texttt{true} : \mathit{fgfrp}{+}\mathit{bndp}{+}\mathit{fgfrp2}{+}\mathit{fgfrp3};$$

however, by taking advantage of the fact that for states/transitions which satisfy multiple guards of a reward structure the reward assigned is the sum of the rewards, often we can write the reward structure in a more readable form. The second reward structure ("*reloc*") assigns a reward of 1 to the transitions associated with FGFR receptors being relocated, i.e. those labelled by *reloc2* or *reloc3*.

We now list a number of properties of the compartments model using CSL and the reward structure given in Figure 13.

- $\mathtt{P}_{=?}[\ \mathtt{true}\ \mathtt{U}\ (\mathit{fgfr2}{+}\mathit{fgfp2}{\geq}l)\ ]$ - 'the probability that at any time there are more than l FGFR receptors being recycled';
- $(\mathit{degp3}{+}\mathit{fgfp3}{=}0) \Rightarrow \mathtt{P}_{\leq 0.01}[\ (\mathit{fgfr2}{+}\mathit{fgfp2}{=}0)\ \mathtt{U}^{[0,t]}\ (\mathit{degp3}{+}\mathit{fgfp3}{\geq}l)\ ]$ - 'if nothing is degraded, then the probability that the number of degraded FGFR receptors reaches l before a receptor enters the recycling compartment is at most 0.01';
- $\mathtt{R}\{\text{"}\mathit{phos}\text{"}\}_{=?}[\ \mathtt{I}^{=t}\ ]$ - 'the expected number of FGFR receptors phosphorylated at t seconds';
- $\mathtt{R}\{\text{"}\mathit{reloc}\text{"}\}_{=?}[\ \mathtt{C}^{\leq t}\ ]$ - 'the expected number of relocations in the first t seconds'.

In Figure 14 we have presented the expected number of phosphorylated FGFR receptors in the main compartment at time t as the probability of moving to recycling when relocated varies. These results correspond to the case when initially the number of FGF ligands equals 50 and the number of FGFR receptors equals 10 (i.e. N=50 and M=10). To demonstrate the different results that are



(a) 10 simulation runs (b) 100 simulation runs (c) model checking

Fig. 14. Expected amount of phosphorylated FGFR in main compartment at time t

obtained through simulation as opposed to model checking, in Figures 14(a) and 14(b) we present the results obtained with PRISM's simulator when averaging over 10 and 100 runs respectively and Figure 14(c) presents the same results when using model checking. The graphs show, that increasing the chance of being relocated to the recycling compartment as opposed to the degradation compartment will increase the amount of phosphorylated FGFR receptors in the main compartment. This is due to the fact that as FGFR receptors from the main compartment relocated to the degradation compartment will not return to the main compartment while those relocated to the recycling compartment will eventually return to the main compartment.

Consider the difference between the plots, we see a large fluctuation in the amount of phosphorylated FGFR receptors in the main compartment when only a small number of runs are considered (Figure 14(a)) and as the number of runs increases these fluctuations diminish (Figure 14(b)), while employing the model checking approach we obtain 'smooth' curves (Figure 14(c)). This can be attributed to the fact that, when we consider an individual run, a reaction (e.g. a binding, phosphorylation or relocation) occurs (with probability 1) at a specific time point and therefore its influence can be seen at that specific time point, while in model checking, an average over all possible runs is considered, and hence the probability of the reaction occurring at a certain time point is also taken into account.

5.2 Exercises

1. Based on the model in Figure 12 and the reward structures in Figure 13, write CSL specifications for the following properties:
   (a) 'the probability that eventually all FGFR receptors get degraded';
   (b) 'if there are l_1 free FGF ligands and l_2 free FGFR receptors in the main compartment, the probability that the first degradation of an FGFR receptor occurs after time t is less than 0.1';
   (c) 'if in the main compartment there are more than l FGF:FGFR compounds phosphorylated, then the expect number of relocations occurring by time t is at least 5.6'.

2. Construct an appropriate reward structure for calculating the expected number of dephosphorylations that occur in the recycling and degradation compartments of the model in Figure 12 and write a CSL specification for the expected number of dephosphorylations in the recycling and degradation compartments by time t.
   Hint: You will need to add extra action labels. Make sure sure that these are all distinct to avoid unwanted synchronisations between modules.
3. Extend the model in Figure 12 with variables to count the number of receptors that get relocated to the recycling compartment and the number of receptors that return from the recycling compartment to the main compartment. Write CSL specifications for the following properties:
   (a) 'if l_1 receptors have entered the recycling compartment and nothing has returned to the main compartment then, with probability at least 0.55, l_2 receptors will return within the next t seconds';
   (b) 'the probability that l_1 receptors return from the main compartment before the total number of relocations is l_2'.
   Hint: Since nothing leaves the degradation compartment you can use the variables in this compartment to determine the number of relocations to this compartment.
4. Rewrite the PRISM description of Figure 12 using a single PRISM module. Check that the new model has the same number of states and transitions as the original. In addition, using the simulation engine, generate graphs similar to those presented in Figures 14(a) and 14(b).

6 Related Work

In [8], PRISM has been used to study a more detailed model of the FGF (Fibroblast Growth Factor) signalling pathway. The model corresponds to a single instance of the pathway, i.e. there can be at most one of each molecule or species. This has the advantage that the resulting state space is relatively small, however the model is still highly complex due to the large number of different interactions that can occur in the pathway and is sufficiently rich to explain the roles of the components in the pathway and how they interact. In [4], PRISM is used to model the RKIP-inhibited ERK pathway using an approximate 'population' approach to modelling in which concentrations are modelled by discrete abstract quantities. Also modelling the RKIP-inhibited ERK pathway, [3] demonstrates how the stochastic process algebra PEPA [9] can be used to model biological pathways. The stochastic π-calculus [18] has also been proposed as a model language for biological systems [20, 19]; this approach has so far been used in conjunction with stochastic simulation, for example through the tools BioSpi [19] and SPiM [16]. A translation from the stochastic π-calculus to PRISM has also been developed [14].

An alternative is to use the language SBML [11], a computer-readable language based on XML for representing models of biochemical reaction networks, and the translator from SBML to the PRISM modelling language [23]. SBML is intended as a standardised representation of models that can be shared, manipulated and analysed using tools available in the systems biology community. Models are composed from components, which permit definition of reactant species, product species, descriptions of reaction equations using MathML expressions, and the specification of kinetic laws and parameters.

The principal challenge remaining for the application of probabilistic model checking to biological systems, as in so many other domains, is the scalability of the techniques to ever larger systems and models. There is hope that some of the techniques that have already been developed in the field of formal verification, such as symmetry reduction, bisimulation minimisation and abstraction, will prove beneficial in this area. For further details on such approaches and pointers to related work, see for example [8].

Acknowledgements

This work was supported in part by the EPSRC grants EP/D076625/2 and EP/D07956X/2. We would also like to thank John Heath (School of Biosciences, University of Birmingham) and Oksana Tymchyshyn (School of Computer Science, University of Birmingham) for their help in developing the examples.

References

1. A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Model checking continuous time Markov chains. *ACM Transactions on Computational Logic*, 1(1):162–170, 2000.
2. C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE Transactions on Software Engineering*, 29(6):524–541, 2003.
3. M. Calder, S. Gilmore, and J. Hillston. Modelling the influence of RKIP on the ERK signalling pathway using the stochastic process algebra PEPA. *Transactions on Computational Systems Biology*, 7:1–23, 2006.
4. M. Calder, V. Vyshemirsky, D. Gilbert, and R. Orton. Analysis of signalling pathways using continuous time Markov chains. *Transactions on Computational Systems Biology*, 4:44–67, 2006.
5. E. Clarke, E. Emerson, and A. Sistla. Automatic verification of finite-state concurrent systems using temporal logics. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.
6. D. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25):2340–2361, 1977.
7. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.
8. J. Heath, M. Kwiatkowska, G. Norman, D. Parker, and O. Tymchyshyn. Probabilistic model checking of complex biological pathways. *Theoretical Computer Science*, 319(3):239–257, 2008.
9. J. Hillston. *A Compositional Approach to Performance Modelling.* Cambridge University Press, 1996.
10. A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In H. Hermanns and J. Palsberg, editors,

Proc. 12th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06), volume 3920 of *Lecture Notes in Computer Science*, pages 441–444. Springer, 2006.

11. M. Hucka, A. Finney, B. Bornstein, S. Keating, B. Shapiro, J. Matthews, B. Kovitz, M. Schilstra, A. Funahashi, J. Doyle, and H. Kitano. Evolving a lingua franca and associated software infrastructure for computational systems biology: The Systems Biology Markup Language (SBML) project. *Systems Biology*, 1(1):41–53, 2004.
12. M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. *Int. Journal on Software Tools for Technology Transfer (STTT)*, 6(2):128–142, 2004.
13. M. Kwiatkowska, G. Norman, and D. Parker. Stochastic model checking. In M. Bernardo and J. Hillston, editors, *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07)*, volume 4486 of *Lecture Notes in Computer Science (Tutorial Volume)*, pages 220–270. Springer, 2007.
14. G. Norman, C. Palamidessi, D. Parker, and P. Wu. Model checking probabilistic and stochastic extensions of the π-calculus. *IEEE Transactions on Software Engineering*, 2009.
15. D. Parker. *Implementation of Symbolic Model Checking for Probabilistic Systems*. PhD thesis, University of Birmingham, 2002.
16. A. Phillips and L. Cardelli. Efficient, correct simulation of biological processes in the stochastic π-calculus. In M. Calder and S. Gilmore, editors, *Proc. Int. Conf. Computational Methods in Systems Biology (CMSB'07)*, volume 4695 of *Lecture Notes in Bioinformatics*, pages 184–199. Springer Verlag, 2007.
17. PRISM web site. http://www.prismmodelchecker.org/.
18. C. Priami. Stochastic π-calculus. *The Computer Journal*, 38(7):578–589, 1995.
19. C. Priami, A. Regev, W. Silverman, and E. Shapiro. Application of a stochastic name passing calculus to representation and simulation of molecular processes. *Information Processing Letters*, 80:25–31, 2001.
20. A. Regev, W. Silverman, and E. Shapiro. Representation and simulation of biochemical processes using the π-calculus process algebra. In R. Altman, A. Dunker, L. Hunter, and T. Klein, editors, *Pacific Symposium on Biocomputing*, volume 6, pages 459–470. World Scientific Press, 2001.
21. W. Roscoe. *Theory and Practice of Concurrency*. Prentice Hall, 1998.
22. J. Rutten, M. Kwiatkowska, G. Norman, and D. Parker. *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems*, volume 23 of *CRM Monograph Series*. AMS, 2004.
23. SBML-to-PRISM translator. http://www.prismmodelchecker.org/sbml/.
24. O. Wolkenhauer, M. Ullah, W. Kolch, and K.-H. Cho. Modeling and simulation of intracellular dynamics: choosing an appropriate framework. *IEEE Transactions on Nanobioscience*, 3:200–207, 2004.